Effects of Large Numerical Models in Continuous Hybrid Simulation

Ferran Obón Santacana & Uwe E. Dorka
University of Kassel, Germany)

SUMMARY:
The field of hybrid simulation has been limited to use fairly simple numerical models (consisting maybe of an order of ten Dynamic Degrees of Freedom DDOF), since the largest and most complex ones were out of reach mainly due to hardware limitations. The presence of multi-core desktop computers and the use of high performance computers together with the still in development European Platform for Geographically Distributed Seismic Tests can increase the number of available DOF for continuous and also real time hybrid simulation. This paper gives an overview of the ongoing work within SERIES to assess the extensibility of such platform to large and complex numerical models.

Keywords: Hybrid Simulation, Sub-structure Testing, Large Numerical Models, Sub-structure algorithm.

1. INTRODUCTION

The increase in computational power of desktop computers has broadened the horizons of hybrid simulation. By using parallel approaches when programming the sub-structure algorithm, in this case the sub-structure algorithm developed by Dorka (Roik (1989) and Dorka (1991, 1998, 2002, 2006, 2007), larger numerical models can be used not only in *continuous* hybrid simulation but also in *real-time* tests.

However, the largest models consisting of several thousands of dynamic degrees of freedom are still out of reach since it would still require too large step times in order to integrate them in continuous tests. The solution is conceptually easy: use more powerful computers; that is a supercomputer or a Linux cluster. Unfortunately, only a handful of laboratories would be able to satisfy the computational requirements on-site. This is where the still in development European Platform for Geographically Distributed Seismic Tests within the SERIES (2012) project comes into play. By combining the different ingredients; a parallel programming approach, high performance computers and distributed testing, much larger models could be used in the context of hybrid simulation.

2. DORKA’S SUB-STRUCTURE ALGORITHM

One of the key components of the previously commented mixture for using large numerical models in *continuous* or even *real-time* hybrid simulation is the substructure algorithm developed by Dorka. It has been successfully applied to a wide variety of situations, both from the civil engineering point of view and the aerospace engineering field as shown in Bayer (2005).

2.1. Basic Formulation

The underlying layer of the substructure algorithm is a discrete formulation of the dynamic equilibrium equation, which is given by:

$$M\frac{d^2u}{dt} + C\frac{du}{dt} + Ku = p(t) + f_r + f_s \tag{2.1}$$

Where M, C and K are respectively the mass, damping and stiffness matrices of the numerical model and u and $p(t)$ are the displacement and loading vector. The terms of f_r (numerical non-linear restoring force) and f_s (force measured at the interface between numerical model and substructure specimen) take care of the restoring forces.

A general solution of the previous equation can be obtained, if a finite element discretisation in the time domain and shape functions with three supporting points[1] (3-step scheme) as in Zienkiewicz (1977) are used. It is worth to point out that nearly all major time stepping algorithms can be obtained, if different weighting functions are used when performing the time integration. If the displacement is discretised in the time domain, Eqn. 2.1 results into:

$$u^{n+1} = K_e^{-1}\cdot\left\{f_{eff}^{n} + f_*{}^{n} + f_e{}^{n}\right\} \tag{2.2}$$

$$K_e^{-1} = K + a_0M + a_1C \tag{2.3}$$

$$f_{eff}^{n} = M(a_0u^n + a_2\dot{u}^n + a_3\ddot{u}^n) + C(a_0u^n + a_4\dot{u}^n + a_5\ddot{u}^n) \tag{2.4}$$

Where $f_* = f_r + f_s + p$, u is the discretised displacement vector in the time domain, the term K_e is the effective stiffness matrix, f_{eff} the effective load vector and f_e is the error force (see 2.1.2). The super-index n is used to differentiate the time steps. The abbreviations introduced are:

$$a_0 = \frac{1}{\beta\Delta t^2};\ a_1 = \frac{\gamma}{\beta\Delta t};a_2 = \frac{1}{\beta\Delta t};\ a_3 = \frac{1}{2\beta} - 1;\ a_4 = \frac{\gamma}{\beta} - 1;\ a_5 = \frac{\Delta t}{2}\left(\frac{\gamma}{\beta} - 2\right)$$

The different parameters such as time intervals, stability, accuracy and numerical damping are analysed in a deeper way in Dorka (2002). It can be concluded that the only unconditionally stable implicit 3-step algorithm without artificial damping and with the least numerical softening is the so-called *Newmark-β*, with $\gamma = 0{,}5$ and $\beta = 0{,}25$.

2.1.1. Sub-stepping

In Eqn. 2.2, there is also the term f_s^{n+1} (inside f_*^{n}) which takes care of the reaction of the subsystem. Since it is only available through measurement, the use of iteration methods will lead to high frequency oscillations during a test. To deal with this problem, a sub-stepping approach has been used. In Dorka (2002) the next equation, Eqn. 2.5, the implicit algorithm is expressed as a form of a general linear control equation within each time step.

$$u^{n+1} = u_0{}^{n+1} + K_e{}^{-1}(f_s^{n+1} + f_r^{n+1}) \tag{2.5}$$

$$u_0^{n+1} = K_e^{-1}\cdot\left\{f_{eff}^{n} + p^{n+1} + f_e{}^{n}\right\} \tag{2.6}$$

Where u_0 is the initial vector (updated at the beginning of the time step) and f_r, f_s the calculated and measured force vectors respectively. Applying a sub-stepping technique, the displacements present in Eqn. 2.5 are introduced to the specimen in small increments as specified in the ramp function, forming the inner loop in Fig. 2.1b:

[1] The formulation can also be extended to four supporting points.

$$x^t = x^{t-\Delta t}\left(1-\frac{i}{k_{sub}}\right)+x^t\left(\frac{i}{k_{sub}}\right) \tag{2.7}$$

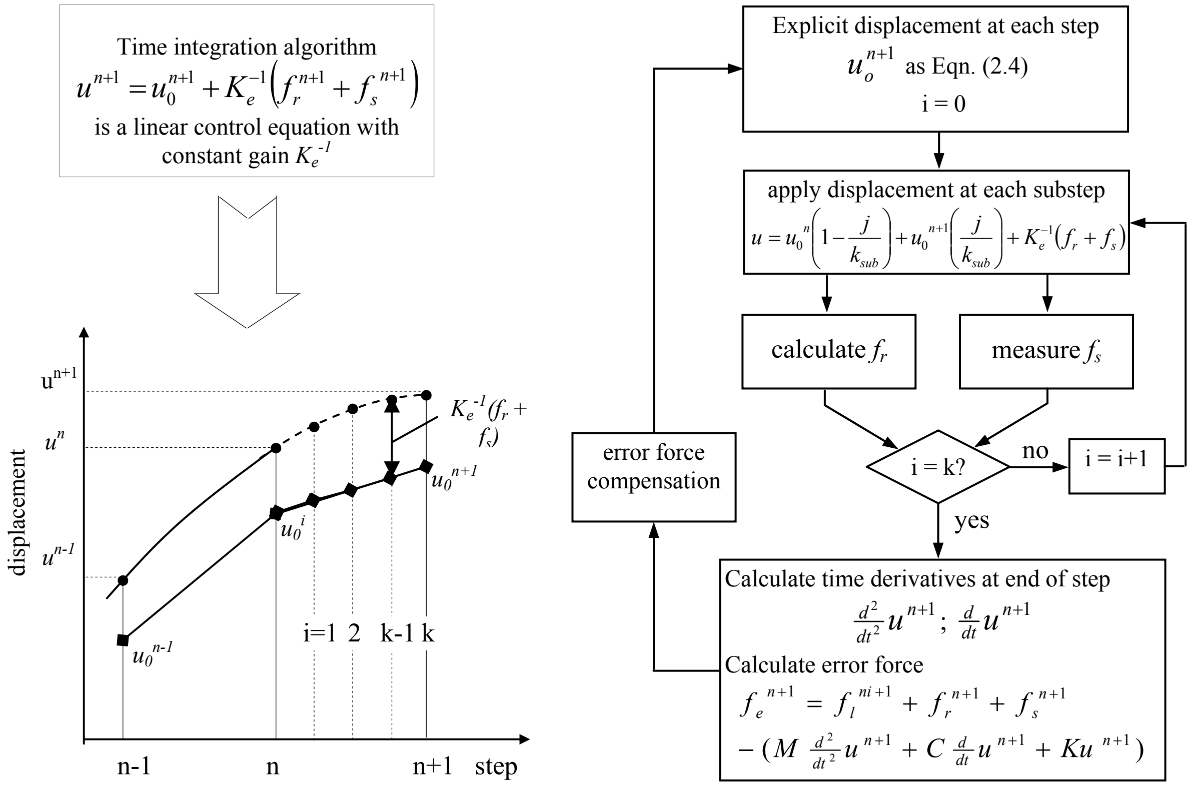
Where k_{sub} is the number of sub-steps and i denotes the current sub-step. In this way, for each sub-step i, the displacement of the subsystem is computed, applied to the test and then the measured or the calculated non-linear restoring forces are given as an input to the algorithm.

2.1.2. Error force compensation

The number of sub-steps k_{sub} (Fig. 2.1a) is an important factor for stability and accuracy (when $k_{sub} \rightarrow \infty$ Eqn. 2.5 and Eqn 2.6 give the exact value), but there are also other sources like positioning error of the actuators, that can make the test unstable. In order to minimise their effect and their tendency to destabilise the test, a PID compensator is used to minimise the error:

$$f_e^n = -P\left[e^n + I\Delta t\sum_i^n e^n + \frac{D}{\Delta t}(e^n - e^{n-1})\right] \tag{2.8}$$

Where e is the equilibrium error defined as the sum of all dynamic forces, whether they are internal or external, at the end of the time step. P, I and D are the adequately chosen proportional, integral and derivative constants of the error compensator, and f_e^n is the compensation force, which is added at the beginning of the next step as a load to the system. Fig. 2.1b summarises how the algorithm runs in displacement control.[2]


Time integration algorithm
$u^{n+1} = u_0^{n+1} + K_e^{-1}\left(f_r^{n+1} + f_s^{n+1}\right)$
is a linear control equation with constant gain K_e^{-1}
displacement
u^{n+1}
u^n
u^{n-1}
$K_e^{-1}(f_r + f_s)$
u_0^{n+1}
u_0^i
u_0^{n-1}
i=1 2 k-1 k
n-1
n
n+1 step
Explicit displacement at each step
u_o^{n+1} as Eqn. (2.4)
i = 0
apply displacement at each substep
$u = u_0^n\left(1-\frac{j}{k_{sub}}\right)+u_0^{n+1}\left(\frac{j}{k_{sub}}\right)+K_e^{-1}(f_r + f_s)$
calculate f_r
measure f_s
i = k?
no
i = i+1
yes
error force compensation
Calculate time derivatives at end of step
$\frac{d^2}{dt^2}u^{n+1}$; $\frac{d}{dt}u^{n+1}$
Calculate error force
$f_e^{n+1} = f_l^{ni+1} + f_r^{n+1} + f_s^{n+1}$
$-(M\frac{d^2}{dt^2}u^{n+1} + C\frac{d}{dt}u^{n+1} + Ku^{n+1})$

(a) – linear control mechanism

(b) – flow chart of substructure algorithm

Figure 2.1. Substructure algorithm developed by Dorka.

[2] It is possible to formulate it also in velocity and acceleration control.

3. PARALLELISATION OF THE SUB-STRUCTURE ALGORITHM

This section will introduce two different approaches of using a parallel version of Dorka's sub-structure algorithm. The first one is using a shared memory approach on regular desktop (or laptop) computers, while the second one is targeted at distributed memory computers (computers with multiple processors connected through a network).

One of the most interesting aspects of this sub-structure algorithm is the possibility to perform the costly operations, like the matrix inversion, before the test starts. This leaves all the operations during a test of the type vector scaling (which is an embarrassingly parallel problem since it requires no communication across multiple processors), vector addition and matrix-vector multiplication.

3.1. Parallelisation on desktop computers

Parallel programming in multi-core computers usually follows the shared memory approach, that is, different processes or cores share the same memory. In this approach, the work is divided among the different cores or processes through a *thread*. There are several libraries that provide this functionality, being Open Multiprocessing (OpenMP, Chapman et. al (2008)) or POSIX Threads (Pthreads) the most popular of them. However, it is not recommended to use these libraries directly if not to do the simplest task or if there is no alternative without a good knowledge on parallel programming. The reasons: lack of experience and knowledge translates into poor performance and it is time consuming.

Some linear algebra operations are so common that several libraries have risen in order to avoid having to program routines such as matrix-vector operations over and over again. Perhaps the most widely adopted of these libraries are the BLAS (Basic Linear Algebra Subroutines) (Lawson (1979)) and LAPACK (Linear Algebra PACKage) (Anderson (1999)). While the first covers vector, matrix-vector and matrix-matrix operations, the second provides means to solve linear systems or matrix inversions and relies upon BLAS for the aforementioned basic operations. Another advantage of using BLAS and LAPACK is that they allow parallelising the program easily. Most of the implementations of such libraries, like ATLAS from Whaley and Petitet (2005), MKL from Intel or OpenBLAS, offer different degrees of parallelisation and the ability to use multiple cores and threads.

Therefore, parallelising the algorithm would simply require the use of multithreaded implementation of such libraries, relegating OpenMP or Pthreads to a minor role. Dealing with TCP/IP communication with the controller or with multiple sub-structures, real or numerical, simultaneously are examples of where OpenMP or Pthreads could be useful. Fig. (3.2) shows a benchmark of the version of the sub-structure algorithm using different BLAS implementations: the single-threaded reference BLAS, ATLAS, OpenBlas and Intel MKL. It is worth saying that the operations performed were done considering symmetrical matrices and single precision (*float* in C) due to the limitations of the controller at the University of Kassel.

As can be seen in the graphs, the implementation of the BLAS library used has a huge impact on performance. The most important information that this figure offers is, however, the approximate limit of the size of the matrices for a given step time in a continuous or a real-time test. It is worth to notice that since the actuator needs to move continuously, the data *must* be at the controller after the last sub-step (to measure the coupling force) but before the beginning of the new sub-step (to move the cylinder without stop). This leaves a window time of $\Delta t/k$ – *communication time*, where Δt is the time step and k is the number of sub-steps. Therefore, for a real-time test the maximum matrix size would be around 2000 for a Intel Core i7 950 (10 *ms* step, 2,5 *ms* sub-step against 2,0 *ms* for computing and 0,4 *ms* communication time with ADwin Jäger (2010), the controller).

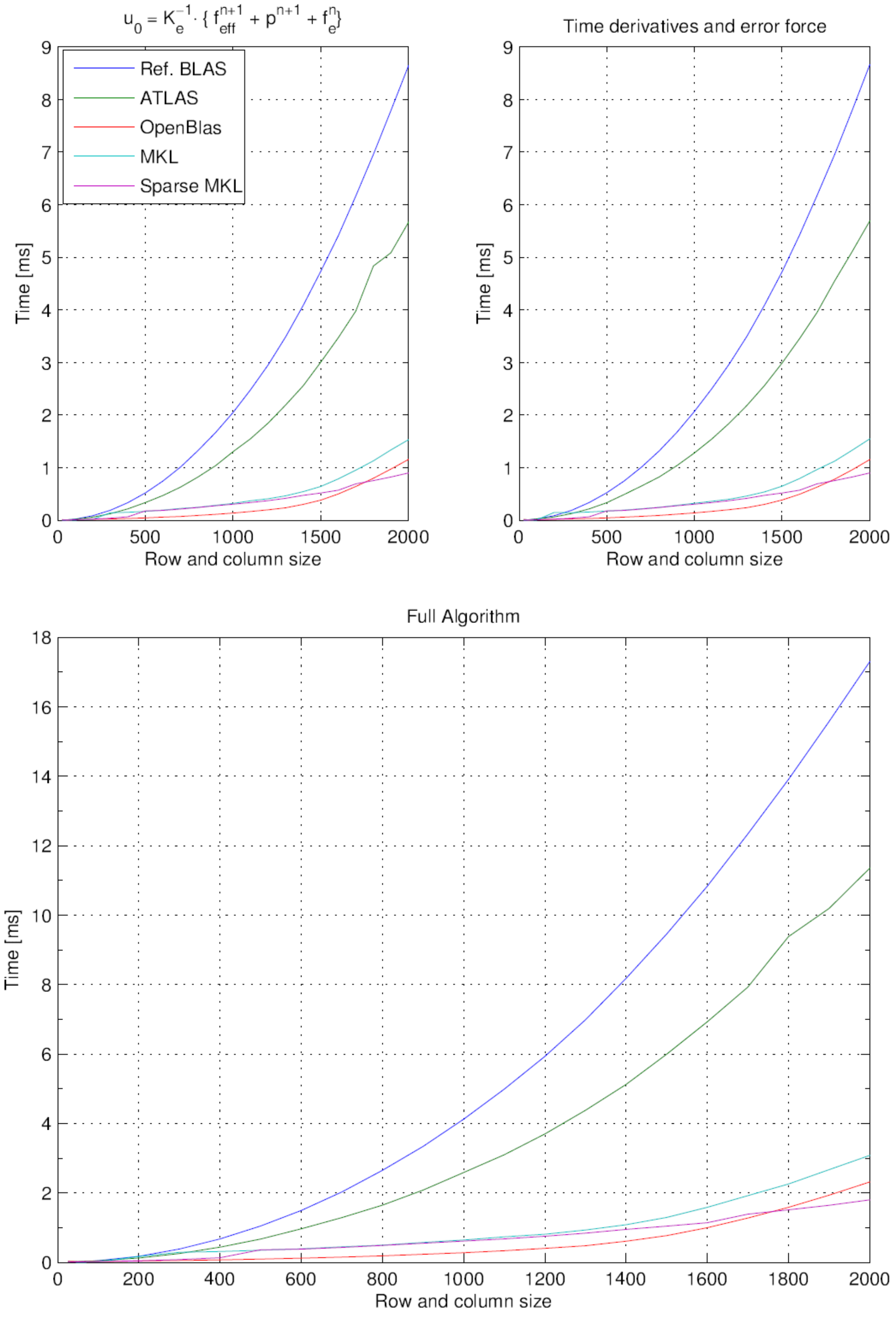


Figure 3.2. Benchmark of the algorithm for different matrix sizes.

It is possible however to increase the performance of the sub-structure algorithm by using sparse operations instead of taking the usual approach of considering dense matrices. Although the improvement would depend then upon the density of the matrices (percentage of non-zero elements), Fig 3.2 can also give an approximate idea of the performance boost using the sparse BLAS implementation of the Intel MKL library and a matrix density of 11%.

3.2. Parallelisation on multiple processors

But what if there are several available processors connected through a network (a cluster or a supercomputer)? BLAS and LAPACK libraries cannot be scaled through multiple nodes (a node is a processor with either a “single” or multiple cores). Since the same applies to OpenMP or Pthreads, other libraries are required. Message Passing Interface (MPI, Gropp et. Al (1999)) fulfils the necessities, but as well as programming with OpenMP the results can be far from satisfying from the performance point of view without some good knowledge on parallel programming.

This is solved through the use of Scalable Linear Algebra PACKage (ScaLAPACK) in Blackfort (1997) and Parallel BLAS (PBLAS) in Chor (1995) and Petitet (1996), which are the linear algebra libraries for distributed memory computers. The communication aspects are also hidden from the end user through the BLACS library (Chor et al (1995), Dongarra and Whaley (1995)). Having more processors available, allows for more aggressive parallelisation of the substructure algorithm. One possible example is the one shown in Fig. 3.1, which features a good load balancing.

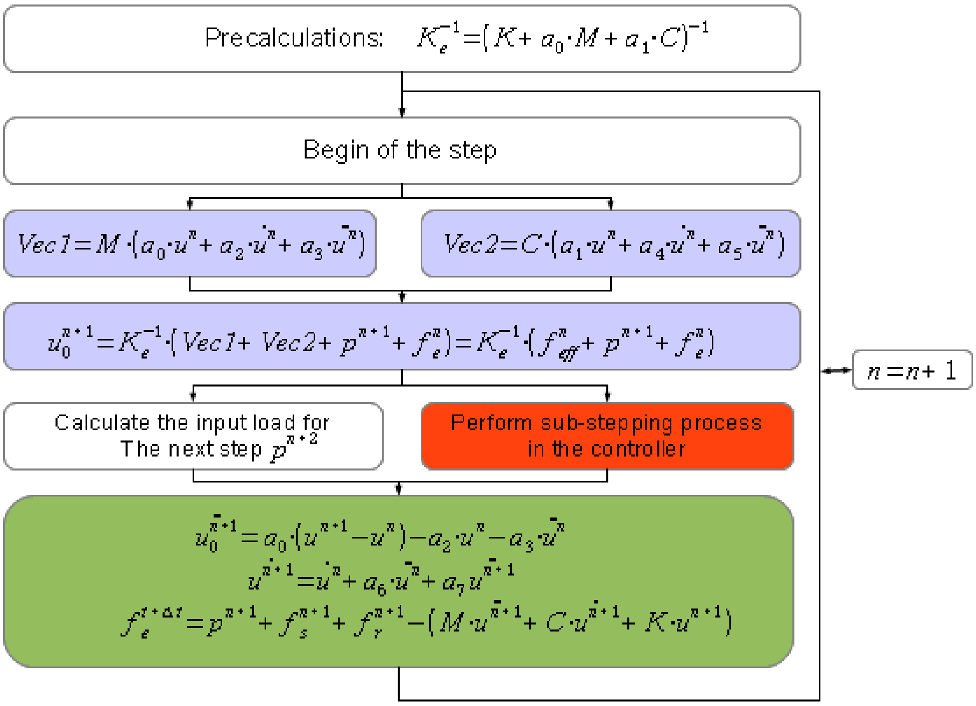

Precalculations: $K_e^{-1}=(K+a_0\cdot M+a_1\cdot C)^{-1}$
Begin of the step
$Vec1=M\cdot(a_0\cdot u^n+a_2\cdot\dot{u}^n+a_3\cdot\bar{u}^n)$
$Vec2=C\cdot(a_1\cdot u^n+a_4\cdot\dot{u}^n+a_5\cdot\bar{u}^n)$
$u_0^{n+1}=K_e^{-1}\cdot(Vec1+Vec2+p^{n+1}+f_e^n)=K_e^{-1}\cdot(f_{eff}^n+p^{n+1}+f_e^n)$
$n=n+1$
Calculate the input load for The next step p^{n+2}
Perform sub-stepping process in the controller
$\bar{u}_0^{n+1}=a_0\cdot(u^{n+1}-u^n)-a_2\cdot u^n-a_3\cdot\bar{u}^n$
$\dot{u}^{n+1}=\dot{u}^n+a_6\cdot\bar{u}^n+a_7\bar{u}^{n+1}$
$f_e^{t+\Delta t}=p^{n+1}+f_s^{n+1}+f_r^{n+1}-(M\cdot\bar{u}^{n+1}+C\cdot\dot{u}^{n+1}+K\cdot u^{n+1})$

Figure 3.1. A possible parallel implementation of Dorka’s substructure algorithm.

4. FURTHER VALIDATION AND TEST SETUP

Different approaches will be used in order to validate both, real-time and continuous distributed sub-structure testing using large numerical models. In the first of them, the real-time approach, a fairly detailed model (504 DOF) of a three-storey steel frame will be used. This non-symmetrical steel frame used during the DFG project “Sub-shake” was initially condensed to 33 DOF in order to be able to run a real time test (Fig 4.1). Due to this, eigenfrequency shifts were introduced starting in the 2nd mode as discussed in Khanlou (2011). It is suspected that this deviations lead to major differences, between the reference and the substructure test, in the frequency domain.

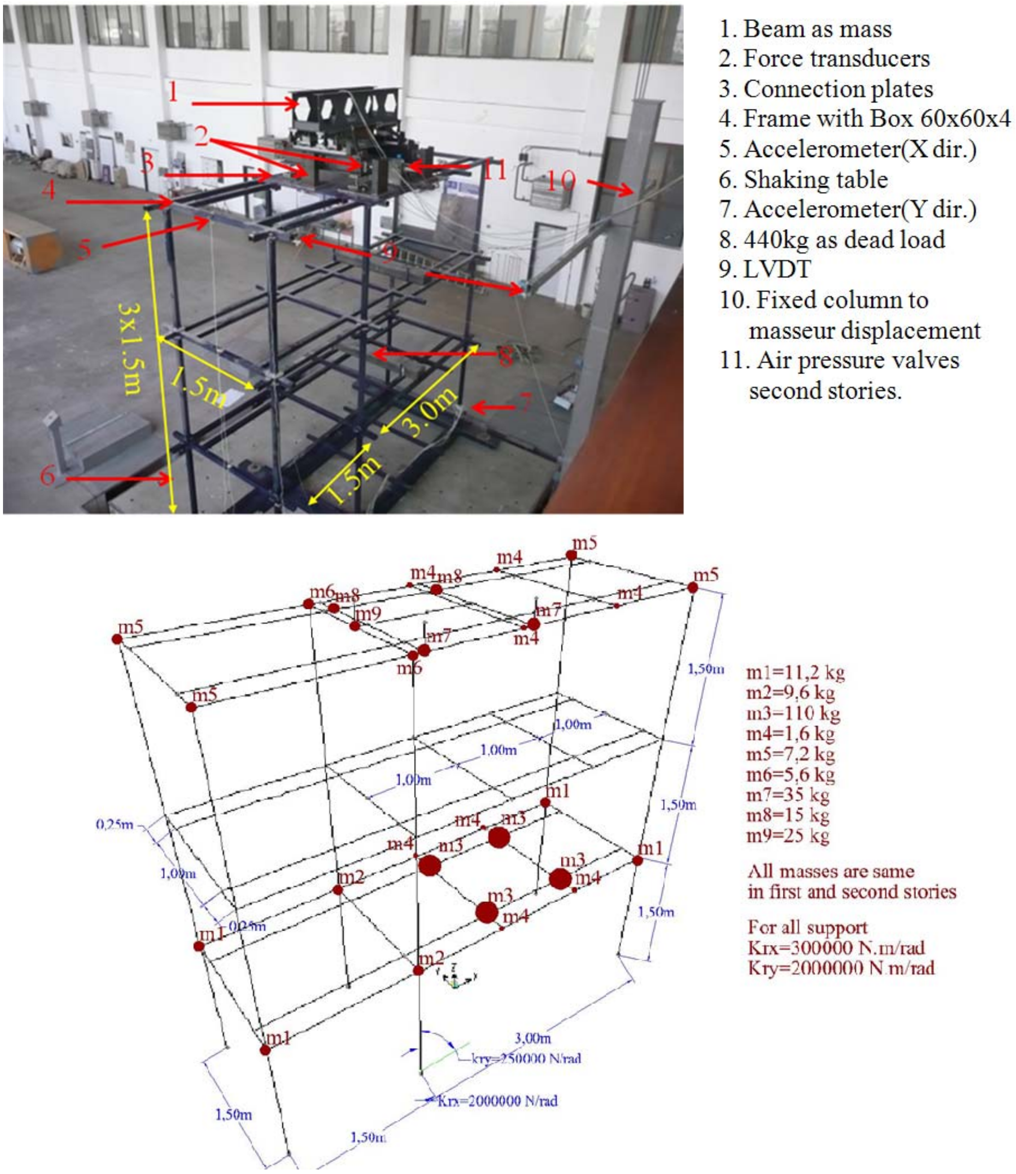


Figure 4.1. Three-storey steel frame with a TMD and a UHYDE-*fbr* (Dorka 1995) on the shake table at IZIIS during the DFG “Sub-shake” project (top) and numerical model (bottom). Pictures from Khanlou et al (2011)

On the sub-structuring side, the shake table in the University of Kassel will be used (Fig 4.2 bottom). Two actuators controlled by ADwin (a system capable of multi-tasking and in charge of the sub-stepping process) will be responsible for introducing the displacements coming from the numerical model. The friction device by Dorka (1995), the UHYDE-*fbr* (Fig 4.2) will be tested in both configurations: no air pressure (making the TMD linear) and with air-pressure (non-linear TMD), where it would dissipate energy through solid friction. This situation would allow comparing the influence of large numerical models in conjunction with both linear and non-linear test set-up using a real specimen as a reference. As can be seen in Fig. 4.2, through the UHYDE-*fbr*, ideal elasto-plastic behaviour can be achieved and through varying the air pressure in the chamber, different hysterises loops can be applied.

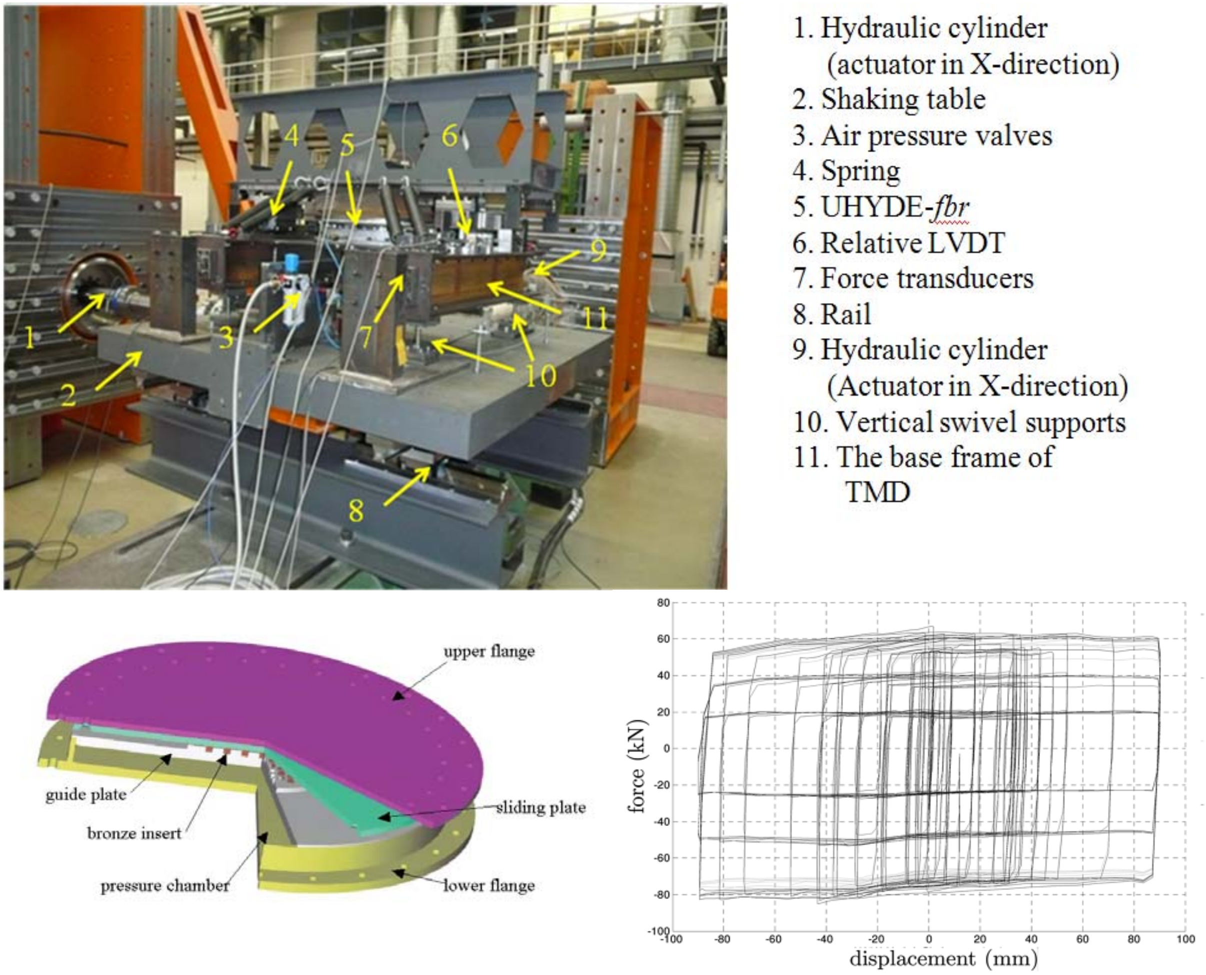


Figure 4.2. Test setup at University of Kassel (top) as in Khanlou et al (2011), concept of the UHYDE-*fbr* (bottom left) as in Dorka (1995), and hysteretic behaviour (bottom right) as in Bossi (2003)

On the geographically distributed continuous test side, much larger models will be used (Fig. 4.3). Two facilities will be involved: the numerical facility (the Linux Cluster at the University of Kassel) and the experimental facility (the shake table in the laboratory). Both facilities have to be in constant communication since for the sub-stepping process, the coupling part of the vector u_0 is required. In order to perform such tests, both the numerical models and the test set-up will need to be modified. In the second case, the TMD would require to be disabled since in time-scaled tests, its dynamic behaviour cannot be captured. This would leave the friction device as the only element in charge of dissipating energy. On the other hand, the numerical models will have the friction-device installed in the concrete cores of the buildings through coupling nodes.

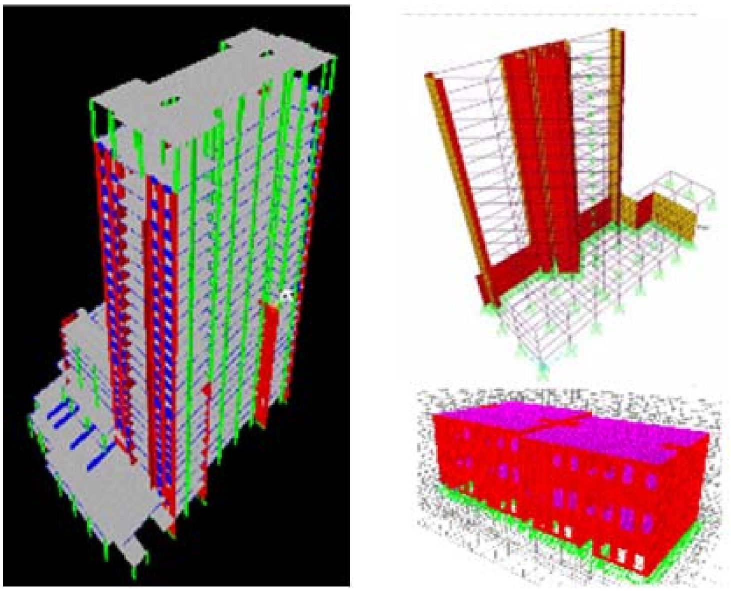


Figure 4.3. On the left: The Contribution of Middle East Technical University and BLA Associates, Atlanta, GA.. On the right: Models contributed by the Technical University of Istanbul. Both in the SERIES project

5. CONCLUSIONS AND FURTHER WORK

The use of large numerical models in both real-time and distributed continuous hybrid simulation will be studied and verified. The use of more detailed numerical models is expected to provide much better accuracy in the results on non-symmetrical structures. One one hand, this is possible in a major part due to the use of Dorka's sub-structure algorithm, which allows running the hydraulic cylinder in a continuous fashion thanks to the approach taken during the sub-stepping process: a linear control equation. On the other hand, the increase of the computational power in common desktop computers and the use of high performance libraries, or the more powerful Linux-clusters or supercomputers that many universities have, allows running large numerical models in hybrid simulation.

While the computing side is mainly finished, there are still some questions that further work will try to address. The influence of higher modes in the results, when more detailed numerical models are needed or what are the parameters that one should control in order to have an acceptable level of detail in the models are possible outcomes of these studies.

AKCNOWLEDGEMENT

This research is part of the SERIES project (Seismic Research Infrastructures for European Synergies) and supported by the European Union under Grant Agreement No. 227887. The authors gratefully acknowledge this support.

REFERENCES

Anderson E., Bai Z., Bischof C., Blackford, S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., and Sorensen D. (1999). *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics.

Bayer V., Dorka U.E., Füllekrug, U. and Gschwilm J. (2005). On real-time pseudo dynamic sub-structure testing: algorithm, numerical and experiment results, *Aerospace Science and Technology*, **Vol.**

9:223-232.

Blackford L.S., Choi J., Cleary A., D'Azevedo E, Demmel J., Dhillon I, Dongarra J., Hammarling S., Henry G., Petitet A, Stanley K., Walker D. and Whaley R.C. (1997). *ScaLAPACK Users' Guide*, Society for Industrial and Applied Mathematics.

Bossi, A. (2003), Isolamento sismico: Prove di Dispositivi con il Metodo Pseudo-dinamico Continuo con Sottostrutturazione, Tesi di Laurea, Universita degli studi di Perugia, Italy, 2003.

Chapman B, Jost G. and Van der Pas R. (2008). Using OpenMP: Portable Shared Memory Parallel Programming, The MIT Press.

Chor J., Dongarra J. Ostroughchov S., Petitet A., Walker D. and Whaley R.C. (1995). A proposal for a set of parallel basic linear algebra subprograms, *LAPACK working note 100*.

Dongarra J. and Whaley R.C. (1995). A user's guide to the BLACS v1.1, *LAPACK working note 94*.

Dorka U.E. (1995), Friction Device for protection of structural systems against dynamic actions, Patent Number 5456047. United States Patent.

Dorka, U.E. and Heiland, D. (1991). Fast online earthquake simulation using a novel pc supported measurement and control concept. *4th Int. Conf. Structural Dynamics*, Southhampton: 636-645.

Dorka, U.E. and Füllekrug, U. (1998). Report of DFG-project No. Do 360/7: Sub-PSD Test. University of Kaiserlautern, Germany.

Dorka U.E. (2002). Hybrid experimental –numerical simulation of vibrating structures, *International Conference WAVE2002*, Okayama, Japan.

Dorka, U.E., Queval, J.C., Nguyen V.T. and Maoult, A.L. (2006). Real-time sub-structure testing on distributed shaking tables in CEA Saclay, *4th World Conference on Structural Control and Monitoring*, San Diego, USA.

Dorka, U.E., Queval, J.C., Nguyen V.T. and Maoult, A.L. (2007). Substructure testing on distributed shaking tables, *2nd International Conference on Advances in Experimental Structural Engineering (2AESE)*, Shanghai, China.

Khanlou, N. M., Dorka, U.E, Ristic, D (2011). Sub-structure test on non-linear TMD using hydraulic shaking table, *International Conference on Earthquake And Structures 2011*, Seoul, South Korea.

Gropp W., Lusk E. and Skjellum A. (1999). Using MPI: Portable Parallel Programming With the Message-passing Intergace, The MIT Press.

Gropp W., Lusk E. and Skjellum A. (1999). Using MPI2: Portable Parallel Programming With the Message-passing Interface, The MIT Press.

Jäger Computergesteuerte Messtechnik GmbH, Rheinstrasse 2-4, D-64653 Lorsch, Germany 2010.

Lawson C.L., Hanson R.J., Kincaid D. and Krogh F.T. (1979). Basic linear algebra subprograms for fortran usage, Vol **5**:308-325.

Petitet A. (1996). Algorithmic Redistribution Methods for Block Cyclic Decompositions, Dissertation, University of Tennessee.

Roik, K. and Dorka, U.E. (1989). Fast online earthquake simulation of friction damped systems, SFB151 Report No. 15, Ruhr-University Bochum, Germany.

SERIES (2012). http://www.series.upatras.gr/ (last visited, April 2012).

Whaley, R. C. and Petitet A. (2005), Minimizing development and maintenance costs in supporting persistently optimized BLAS, Software: Practice and Experience, Vol 35:2:101-121.

Zienkiewicz O.C. (1977). *The Finite Element Method*, McGraw-Hill and Maidenhead.